PRECONDITIONERS AND TENSOR PRODUCT SOLVERS FOR OPTIMAL CONTROL PROBLEMS FROM CHEMOTAXIS

PRECONDITIONERS AND TENSOR PRODUCT SOLVERS FOR OPTIMAL CONTROL PROBLEMS FROM CHEMOTAXIS

SERGEY DOLGOV* AND JOHN W. PEARSON†

Abstract. In this paper, we consider the fast numerical solution of an optimal control formulation of the Keller–Segel model for bacterial chemotaxis. Upon discretization, this problem requires the solution of huge-scale saddle point systems to guarantee accurate solutions. We consider the derivation of effective preconditioners for these matrix systems, which may be embedded within suitable iterative methods to accelerate their convergence. We also construct low-rank tensor-train techniques which enable us to present efficient and feasible algorithms for problems that are finely discretized in the space and time variables. Numerical results demonstrate that the number of preconditioned GMRES iterations depends mildly on the model parameters. Moreover, the low-rank solver makes the computing time and memory costs sublinear in the original problem size.

Key words. PDE-constrained optimization; Boundary control; Preconditioning; Chemotaxis; Mathematical biology

AMS subject classifications. 35Q93, 65F08, 65F10, 65N22, 92C17

1. Introduction.

The process of chemotaxis in biology describes the movement of cells or organisms in a directed fashion as a response to external chemical signals. In 1971, Keller and Segel presented a mathematical model for bacterial chemotaxis [23]. In essence, for large numbers of bacteria, it is predicted that the bacteria will on average move up gradients of the chemoattractant concentration.

Since Keller and Segel's work, an area of numerical mathematics that has become a subject of significant interest is that of PDE-constrained optimization, where one wishes to predict the circumstances in which some physical (or in this case biological) objective occurs, subject to a system of PDEs describing the process. Using this technology, one is able to pose an optimal control problem for the chemotaxis mechanism: given an observed bacterial cell concentration profile, what can be said about the external chemoattractant at the boundaries of a domain of interest? The constraints for this problem are therefore the PDEs describing bacterial chemotaxis. This is a parameter identification problem that has been considered in literature such as [28, 46], and in particular it was shown numerically by Lebiedz and Brandt-Pollmann that "it is possible to systematically control spatiotemporal dynamical behavior" [28].

The fast and efficient iterative solution of PDE-constrained optimization problems has increasingly become an active area of research, and in particular it is now widely recognized that the incorporation of effective preconditioners to accelerate iterative schemes is highly beneficial from a computational point of view. Preconditioning theory and numerics for a number of steady [42, 43, 44, 47, 54, 67] and time-dependent [6, 40, 41, 56] problems have been established, with [40, 56] describing the resulting solvers for reaction–diffusion problems from chemistry and biology. In this paper, we derive a potent preconditioner for the chemotaxis problem based on the saddle point structure of the matrix systems resulting from Newton-type iterations of the nonlinear PDEs.

When solving these optimization problems, which often involve the solution of a system of PDEs with initial conditions coupled with adjoint PDEs equipped with

*Department of Mathematical Sciences, University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom (s.dolgov@bath.ac.uk)

†School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom (j.pearson@ed.ac.uk)

final-time conditions, there are many challenges arising from the time-dependent component of the problem in particular, due to the forward-backward solves required, and the associated scaling of computational complexity with the fineness of the grid in the time variable. Difficulties also arise from nonlinear problems, due to the matrices arising from the PDE system varying in structure at every time step, unlike linear problems for which some matrices can be re-used repeatedly within a solver. For time-dependent nonlinear problems that arise from chemotaxis, computer storage is therefore a significant bottleneck, unless a numerical algorithm is specifically tailored in order to mitigate this.

To combat this issue, in addition to presenting our new preconditioner, we describe an approach for approximating the solution of our problem in a low-rank format, namely the Tensor Train (TT) decomposition [37]. Low-rank tensor techniques emerge from the separation of variables and the Fourier method for solving PDEs. We can approximate the solution in the form $z(x,y) \approx \sum_{i=1}^{r} v_i(x) g_i(y)$, using a possibly small number of terms r. In this case, the discretized univariate functions in the low-rank decomposition are much cheaper to store than the original multivariate function. The discretized separation of variables requires the low-rank approximation of matrices (for two variables x, y), or tensors (for three or more variables). Direct generalization of the separation of variables to three and more dimensions was called the Canonical or CP decomposition [26]. However, it may suffer from the ill-posedness and the instability of the approximation problem [5]. TT and, more generally, hierarchical tensor decompositions, such as HT [15], consist of recurrent matrix factorizations, and have a more complicated summation structure than CP. However, this allows us to employ robust tools of linear algebra, such as the singular value decomposition, to deliver an optimal low-rank approximation for a desired accuracy. Moreover, iterative algorithms exist for efficient solution of linear equations that compute directly the decomposition factors of the approximate solution. This gets rid of the excessive numerical cost of solving the full problem, in contrast to the offline stage in reduced basis methods. Further details are provided in Section 6, and general reviews on the topic can be found in [13, 14, 25].

The efficiency of low-rank decompositions depends crucially on the value of the rank r, which in turn reflects the structure of a function. The dimension of a subspace needed to approximate an arbitrary function with bounded weak partial derivatives up to order s with an error ε in the L^2 norm is proportional to $\varepsilon^{-d/s}$. This exponential growth of the cost with d is called the curse of dimensionality, which plagues all functions of low regularity. Discontinuous functions, in particular level set functions, are thus extremely difficult for any numerical representation based on linear subspaces. However, bounding also the *mixed* derivatives up to order s, and decomposing the coefficients in a hierarchical tensor format, reduces the complexity estimate to $\mathcal{O}(\varepsilon^{-3/(2s)} |\log \varepsilon|^{d-2})$ for $s \gg 1$ [53, Thm. 2]. The optimal control problem implies driving the solution to a desired state, which often has a simple structure and hence is usually highly regular. Therefore, as long as we avoid discontinuous functions in our formulation, the low-rank techniques can be very efficient for the optimal control problem. This is demonstrated in our computational experiments.

This paper is structured as follows. In Section 2 we describe the problem statement of which the numerical solution is considered. In Section 3 we present the structure of the matrix systems that result from the discretization of the system of PDEs. In Section 4 we present our preconditioning strategy for these systems, with numerical results provided in Section 5. We describe the low-rank tensor decomposi-

tion which is employed for the matrix systems in Section 6, with additional numerical experiments relating to this approach carried out in Section 7. Finally, concluding remarks are made in Section 8.

2. Problem statement. We base our investigation on optimal control modelling of chemotaxis problems considered in literature such as [28] and [46, Ch. 13]. These works are inspired by the original Keller–Segel model for chemotaxis [23], and investigations by Tyson et al. deriving 'forward' PDE formulations for bacterial chemotaxis [60, 61]. Numerical methods have previously been derived for the forward problem, and to a lesser extent to optimal control analogues, for instance finite difference methods [28, 29, 50], finite element methods [30, 50, 57, 58], and finite volume methods [62]. An incomplete LU preconditioner was implemented for a finite element discretization in [30], applied within the BiCGSTAB method [64]. However, to our knowledge, neither preconditioned iterative methods nor tensor product solvers have been derived for optimal control problems from chemotaxis – in this paper we examine each of these new challenges.

We commence by examining the following problem describing the optimal control of a bacterial chemotaxis system, built around the studies of [28, 46]:

$$\min_{z,c,u} \quad \frac{1}{2}\int_{\Omega}(z(\mathbf{x},T)-\widehat{z})^2+\frac{\gamma_c}{2}\int_{\Omega}(c(\mathbf{x},T)-\widehat{c})^2+\frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)}u^2 \tag{2.1}$$

subject to

$$\begin{aligned}\frac{\partial z}{\partial t}-D_z\nabla^2 z+\alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)&=0 \qquad \text{on } \Omega\times(0,T),\\ \frac{\partial c}{\partial t}-\nabla^2 c+\rho c-w\frac{z^2}{1+z^2}&=0 \qquad \text{on } \Omega\times(0,T),\end{aligned}$$

equipped with the boundary conditions and initial conditions:

$$\begin{aligned}D_z\frac{\partial z}{\partial n}-\alpha\frac{z}{(1+c)^2}\frac{\partial c}{\partial n}&=0 \qquad \text{on } \partial\Omega\times(0,T),\\ \frac{\partial c}{\partial n}+\beta c&=\beta u \qquad \text{on } \partial\Omega\times(0,T),\\ z(\mathbf{x},0)&=z_0(\mathbf{x}) \quad \text{on } \Omega,\\ c(\mathbf{x},0)&=c_0(\mathbf{x}) \quad \text{on } \Omega,\end{aligned}$$

with the first condition posed to ensure conservation of mass on the boundary. This problem is solved on a space-time domain $\Omega\times(0,T)$ with boundary $\partial\Omega\times(0,T)$, and for $\Omega\subset\mathbb{R}^2$. The variables z, c denote *state variables*, corresponding to the bacterial cell density and chemoattractant concentration respectively, with u the *control variable*, $\widehat{z}$, $\widehat{c}$ given *desired states* at time T, z_0, c_0 given initial conditions, and γ_c, γ_u, D_z, α, ρ, w, β given (nonnegative) parameters. Following the work of [28, 46], values $D_z = 0.33$, $\alpha = 80$, $\rho = 0$, $w = 1$, $\beta = 0.1$ are chosen for our experiments. We highlight that, by construction of the problem, the control u in some sense relates to the gradient of chemoattractant concentration on the boundary of the domain of interest. The form of the boundary condition which enforces the control makes this a *boundary control problem*. In this PDE-constrained optimization model, we wish to discover what the

profile of this control must be in order for the biological system to behave in a way prescribed by the desired states $\widehat{z}$, $\widehat{c}$.

REMARK 1. *Although derived similarly, the main difference between the works of [28] and [46] is that [28] considers solely the misfit between z and $\widehat{z}$, regularized by a term involving the final time T. Both works involve control variables imposed through a Neumann boundary condition, but whereas this variable appears directly within the cost functional in [46], this is effectively replaced by a regularization of T in [28]. The problem statement* (2.1) *is motivated strongly by the work of [46]. However, as we believe that the methods introduced in this paper could be applied to either cost functional, we wish to highlight that other useful problem formulations are available.*

REMARK 2. *Despite the complex structure of boundary control systems involving chemotaxis equations, it is possible to prove existence of solutions to problems of this form. We highlight the work of [10], where a slightly different form of the problem* (2.1) *with PDE constraints is analyzed (specifically, the author considers terms in the objective functions that measure the misfit between the state variables and desired states within the entire time interval, and with different zeroth order derivative terms within the PDEs). In this setting, assuming a control variable belonging to $L^{\bar{r}}(0,T;L^{\bar{p}}(\partial\Omega))$ in d dimensions (in this paper $d=2$), with $\bar{r}\geq 2$ and $\bar{p}>d$ such that $\frac{2}{\bar{r}}+\frac{d}{\bar{p}}<1$, a unique solution $(z,c)\in\left[L^{\bar{r}}(0,T;W^{\bar{p},1}(\Omega))\cap W^{\bar{r},1}(0,T;W^{\bar{p},-1}(\Omega))\right]^2$ exists. The author shows that the same result holds for the adjoint variables (p,q) provided $\bar{r}>2\bar{p}$.*

We now consider the first and second derivatives of the Lagrangian[1]:

$$\begin{aligned}\mathcal{L}(z,c,u,p,q) = \frac{1}{2}\int_\Omega (z(\mathbf{x},T)-\widehat{z})^2 + \frac{\gamma_c}{2}\int_\Omega (c(\mathbf{x},T)-\widehat{c})^2 + \frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)} u^2 \\ + \int_{\Omega\times(0,T)} p_\Omega\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z + \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\ + \int_{\Omega\times(0,T)} q_\Omega\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \\ + \int_{\partial\Omega\times(0,T)} p_{\partial\Omega}\left(\frac{\partial z}{\partial n}\right) + \int_{\partial\Omega\times(0,T)} q_{\partial\Omega}\left(\frac{\partial c}{\partial n} + \beta c - \beta u\right),\end{aligned}$$

where p and q denote the adjoint variables corresponding to z and c, with p_Ω, q_Ω the components of p, q within the interior of Ω, and $p_{\partial\Omega}$, $q_{\partial\Omega}$ the components on the boundary. We arrive at the following system for the Newton formulations of the

[1]For ease of notation, we exclude initial conditions within the definition of the Lagrangian.

first-order optimality conditions:

$$\begin{aligned}
&\frac{\partial s_z}{\partial t} - D_z\nabla^2 s_z - \alpha\nabla\cdot\left(\nabla\left(\frac{1}{1+c}\right)s_z\right) + \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}s_c\right)z\right) \qquad (2.2)\\
&\qquad = -\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z + \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \quad \text{on } \Omega\times(0,T),\\
&\frac{\partial s_c}{\partial t} - \nabla^2 s_c + \rho s_c - 2w\frac{z}{(1+z^2)^2}s_z \qquad (2.3)\\
&\qquad = -\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \qquad \text{on } \Omega\times(0,T),\\
&\gamma_u s_u - \beta s_q = -\left(\gamma_u u - \beta q\right) \qquad \text{on } \partial\Omega\times(0,T), \qquad (2.4)\\
&-2wq\frac{1-3z^2}{(1+z^2)^3}s_z - \alpha\nabla\left(\frac{2c}{(1+c^2)^2}s_c\right)\cdot\nabla p \qquad (2.5)\\
&\qquad -\frac{\partial s_p}{\partial t} - D_z\nabla^2 s_p + \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla s_p - 2w\frac{z}{(1+z^2)^2}s_q\\
&\qquad = -\left(-\frac{\partial p}{\partial t} - D_z\nabla^2 p + \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right)\\
&\qquad\qquad \text{on } \Omega\times(0,T),\\
&\alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)s_z\right) - \alpha p\nabla\cdot\left(\nabla\left(\frac{2c}{(1+c^2)^2}s_c\right)z\right) \qquad (2.6)\\
&\qquad + \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right)s_p - \frac{\partial s_q}{\partial t} - \nabla^2 s_q + \rho s_q\\
&\qquad = -\left(\alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right)\\
&\qquad\qquad \text{on } \Omega\times(0,T),
\end{aligned}$$

where s_z, s_c, s_u, s_p, s_q are the Newton updates for z, c, u, p, q. The boundary conditions for the updates of the state variables are given by

$$\begin{aligned}
D_z\frac{\partial s_z}{\partial n} - \alpha\frac{s_z}{(1+c)^2}\frac{\partial c}{\partial n} + 2\alpha\frac{zs_c}{(1+c)^3}\frac{\partial c}{\partial n} - \alpha\frac{z}{(1+c)^2}\frac{\partial s_c}{\partial n} = 0 \qquad &\text{on } \partial\Omega\times(0,T),\\
\frac{\partial s_c}{\partial n} + \beta s_c = \beta s_u \quad &\text{on } \partial\Omega\times(0,T).
\end{aligned}$$

Zero initial conditions are specified for s_z, s_c, as well as final-time conditions $s_p(\mathbf{x},T) = -s_z(\mathbf{x},T)$, $s_q(\mathbf{x},T) = -\gamma_c s_c(\mathbf{x},T)$, assuming an initial guess is chosen that satisfies the initial conditions for z, c, and the final-time conditions $p(\mathbf{x},T) = \widehat{z} - z(\mathbf{x},T)$, $q(\mathbf{x},T) = \gamma_c(\widehat{c} - c(\mathbf{x},T))$.

REMARK 3. *We note that Newton iterates for the above first-order conditions can be attracted to maxima or other stationary points of the Lagrangian, although feasible descent directions exist. Whilst we do not encounter this issue within the numerical experiments presented, in this case second-order sufficiency conditions may be derived and verified. We refer the reader to an analysis of second-order conditions for optimal control problems in chemotaxis presented in [10, Ch. 7], and to [3] for a wider discussion of the role of second-order conditions in PDE control problems.*

REMARK 4. *We highlight that there also exist chemotaxis problems which may be written in distributed control form. For example the work in [9], on the identification*

of chemotaxis models with volume-filling, considers (amongst others) a problem which may be interpreted in our setting in the following way:

$$\begin{aligned}
\min_{z,f} \quad & \frac{1}{2}\int_{\Omega\times(0,T)} (z-\widehat{z})^2 + \frac{\gamma}{2}\int_{\Omega\times(0,T)} \left[f^2 + |\nabla f|^2\right] \\
\text{s.t.} \quad & \frac{\partial z}{\partial t} - \nabla^2 z + f\nabla^2 c + \nabla f\cdot\nabla c = 0 && \text{on } \Omega\times(0,T), \\
& -\nabla^2 c + c = z && \text{on } \Omega\times(0,T), \\
& \frac{\partial z}{\partial n} - f\frac{\partial c}{\partial n} = 0 && \text{on } \partial\Omega\times(0,T), \\
& \frac{\partial c}{\partial n} = 0 && \text{on } \partial\Omega\times(0,T), \\
& z(\mathbf{x},0) = z_0(\mathbf{x}) && \text{on } \Omega,
\end{aligned}$$

where γ is a positive constant, and $f(z)$ denotes the chemoattractant sensitivity. The challenge in this case is to discover the necessary profile of the function f in order to drive the chemoattractant to a particular state. We believe that variants of the techniques introduced in this paper could also be applied to this distributed control problem.

3. Matrix systems for Newton and Gauss–Newton. In this section we describe the matrix systems which are obtained by discretization of the optimization problem (2.1). Concatenating the Newton equations (2.2)–(2.6), along with boundary conditions and initial/final-time conditions, gives a block matrix system of the following form:

$$\begin{bmatrix} \mathcal{L}_{zz} & \mathcal{L}_{zc} & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ \mathcal{L}_{cz} & \mathcal{L}_{cc} & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ 0 & 0 & \gamma_u\cdot\mathrm{Id} & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\ \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0 \end{bmatrix} \begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix} \tag{3.1}$$

$$= \begin{bmatrix} \widehat{z} - \left(\chi_{\Omega_T}(z) - \frac{\partial p}{\partial t} - D_z\nabla^2 p + \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right) \\ \gamma_c\widehat{c} - \left(\gamma_c\chi_{\Omega_T}(c) + \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right) \\ -\left(\gamma_u u - \beta q\right) \\ -\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z + \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\ -\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \end{bmatrix},$$

where

$$\begin{bmatrix} \mathcal{L}_{zz} & \mathcal{L}_{zc} \\ \mathcal{L}_{cz} & \mathcal{L}_{cc} \end{bmatrix} = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) - 2wq\frac{1-3z^2}{(1+z^2)^3} & -\alpha\nabla\left(\frac{2c}{(1+c^2)^2}\blacksquare\right)\cdot\nabla p \\ \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)\blacksquare\right) & \gamma_c\chi_{\Omega_T}(\blacksquare) - \alpha p\nabla\cdot\left(\nabla\left(\frac{2c}{(1+c^2)^2}\blacksquare\right)z\right) \end{bmatrix},$$

$$\begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial t} - D_z\nabla^2 - \alpha\nabla\cdot\left(\nabla\left(\frac{1}{1+c}\right)\blacksquare\right) & \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\blacksquare\right)z\right) \\ -2w\frac{z}{(1+z^2)^2} & \frac{\partial}{\partial t} - \nabla^2 + \rho\cdot\mathrm{Id} \end{bmatrix},$$

$$\begin{bmatrix} \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ \mathcal{L}_{cp} & \mathcal{L}_{cq} \end{bmatrix} = \begin{bmatrix} -\frac{\partial}{\partial t} - D_z\nabla^2 + \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla & -2w\frac{z}{(1+z^2)^2} \\ \alpha\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) & -\frac{\partial}{\partial t} - \nabla^2 + \rho\cdot\mathrm{Id} \end{bmatrix},$$

with Id denoting the identity operator, $\chi_{\partial\Omega}(\blacksquare)$ representing a function that restricts the variable to the boundary $\partial\Omega$, and $\chi_{\Omega_T}(\blacksquare)$ similarly denoting a function that restricts the variable to time $t = T$. The equations (3.1) are then discretized using a finite element method, seeking test functions for the state variables within $H^1(\Omega)$, and those for the control variable within $L^2(\Omega)$.

Although it might be possible to devise spectral methods for such control problems, there may be certain difficulties. The resulting matrix systems would be dense, so they ought to be small. However, the accuracy may deteriorate due to a non-continuously differentiable objective function arising from the algebraic constraints on the control variable and due to sharp peaks within the initial states. The low order of accuracy warrants many degrees of freedom, leading to large *and* dense matrices. So we instead proceed with the finite element approach.

As an alternative to solving the Newton system (3.1), it is possible to instead consider a Gauss–Newton approximation, where one neglects second derivatives within the $(1,1)$-block of the saddle point matrix as defined in Section 4. This results in the solution of systems

$$\begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\ \mathcal{L}_{pz} & \mathcal{L}_{qz} & 0 & 0 & 0 \\ \mathcal{L}_{pc} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0 \end{bmatrix} \begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix} = \mathbf{b}, \quad (3.2)$$

where $\mathbf{b}$ is the same right-hand side vector as in (3.1).

To be more explicit about the $\chi_{\Omega_T}(\blacksquare)$ and $\chi_{\partial\Omega}(\blacksquare)$ terms, the associated matrices contain entries of the form $\int_\Omega \phi_i \cdot \phi_j|_{t=T}$ and $\int_{\partial\Omega\times(0,T)} \phi_i \cdot \phi_j|_{\partial\Omega}$ respectively, for finite element basis functions $\{\phi_i\}$ of the same form for each PDE variable.

3.1. Additional control constraints. It is perfectly reasonable to add the following control constraint:

$$u_-(\mathbf{x},t) \le u \le u_+(\mathbf{x},t) \quad \text{a.e. on } \partial\Omega \times (0,T),$$

for given functions u_-, u_+, into the PDE-constrained optimization model, as in [46] for instance. In other words, we prescribe that the chemoattractant must behave in a "sensible" (physical) way on the boundary of the domain of interest. One way in which we can tackle this additional term is to modify the cost functional (2.1) to add a Moreau–Yosida regularization term (see [20]) for the bound constraints, thereby minimizing instead

$$\begin{aligned} \min_{z,c,u} \quad & \frac{1}{2}\int_\Omega (z(\mathbf{x},T) - \widehat{z})^2 + \frac{\gamma_c}{2}\int_\Omega (c(\mathbf{x},T) - \widehat{c})^2 + \frac{\gamma_u}{2}\int_{\partial\Omega\times(0,T)} u^2 \\ & + \frac{1}{2\delta}\int_{\Omega\times(0,T)} |\max\{0, u - u_+\}|^2 + \frac{1}{2\delta}\int_{\Omega\times(0,T)} |\min\{0, u - u_-\}|^2, \end{aligned}$$

with δ a given (small) positive constant, chosen to enforce the control constraints efficiently.

When forming the Newton system in this setting, we will be required to solve

systems relating to the finite element discretization of the following terms:

$$\begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) & 0 & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} + \frac{1}{\delta}G_\Lambda & 0 & -\beta\chi_{\partial\Omega}(\blacksquare)^\top \\ \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) & 0 & 0 \end{bmatrix} \begin{bmatrix} s_z \\ s_c \\ s_u \\ s_p \\ s_q \end{bmatrix} = \widetilde{\mathbf{b}}, \quad (3.3)$$

where

$$\widetilde{\mathbf{b}} := \begin{bmatrix} \widehat{z} - \left(\chi_{\Omega_T}(z) - \frac{\partial p}{\partial t} - D_z\nabla^2 p + \alpha\nabla\left(\frac{1}{1+c}\right)\cdot\nabla p - 2w\frac{zq}{(1+z^2)^2}\right) \\ \gamma_c\widehat{c} - \left(\gamma_c\chi_{\Omega_T}(c) + \alpha p\nabla\cdot\left(\nabla\left(\frac{1}{(1+c)^2}\right)z\right) - \frac{\partial q}{\partial t} - \nabla^2 q + \rho q\right) \\ \frac{1}{\delta}(G_{\Lambda_+}y_+ + G_{\Lambda_-}y_-) - (\gamma_u u - \beta q) \\ -\left(\frac{\partial z}{\partial t} - D_z\nabla^2 z + \alpha\nabla\cdot\left(\frac{\nabla c}{(1+c)^2}z\right)\right) \\ -\left(\frac{\partial c}{\partial t} - \nabla^2 c + \rho c - w\frac{z^2}{1+z^2}\right) \end{bmatrix}.$$

Here, G_{Λ_+}, G_{Λ_-}, G_Λ denote projections onto the active sets $\Lambda_+ := \{i : u_i > (u_+)_i\}$, $\Lambda_- := \{i : u_i < (u_-)_i\}$, $\Lambda := \Lambda_+ \cup \Lambda_-$ (for the i-th node on the discrete level).

For PDE control problems involving additional constraints on the control variable, as above, the stationarity conditions are no longer continuously differentiable, and Newton's method becomes a semi-smooth Newton method. Such methods are well-established for problems involving control constraints [16, 17, 48, 63], including challenging boundary control problems [4]. In particular, convergence of semi-smooth Newton methods for operator equations in function spaces is shown in [63], and [17] discusses mesh-independent convergence.

The application of Newton (including Gauss–Newton) and semi-smooth methods allows us to reformulate the optimal control problems being studied as a sequence of linearized problems. The main challenge is now to establish potent solvers for the matrix systems arising at each Newton iteration, for which we now focus on the development of preconditioned iterative methods.

4. Preconditioning for Gauss–Newton matrix systems. In this section we focus on deriving effective preconditioners for the matrix systems (3.2) and (3.3), resulting from the chemotaxis model, both without and with additional control constraints.

We base our preconditioners on the well-studied field of *saddle point systems*, which take the form [2]

$$\underbrace{\begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \quad (4.1)$$

with A symmetric, and B having at least as many columns as rows. For instance, in the case where A is invertible and B has full row-rank, two well-studied preconditioners for the system (4.1), are given by [19, 27, 32]

$$\mathcal{P}_D = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}, \qquad \mathcal{P}_T = \begin{bmatrix} A & 0 \\ B & -S \end{bmatrix},$$

where the (negative) *Schur complement* $S := BA^{-1}B^\top$. It is known [19, 27, 32] that, as the preconditioned system is nonsingular, its eigenvalues are given by

$$\lambda(\mathcal{P}_D^{-1}\mathcal{A}) \in \left\{1, \frac{1}{2}(1 \pm \sqrt{5})\right\}, \qquad \lambda(\mathcal{P}_T^{-1}\mathcal{A}) \in \{1\},$$

with these results also holding for the block triangular preconditioner $\mathcal{P}_T$ even if A is not symmetric. Now, as $\mathcal{P}_D^{-1}\mathcal{A}$ is diagonalizable but $\mathcal{P}_T^{-1}\mathcal{A}$ is not, preconditioning with $\mathcal{P}_D$ ($\mathcal{P}_T$) yields convergence of a suitable Krylov subspace method in 3 (2) iterations, respectively. We note that in practice, however, $\mathcal{P}_D$ and $\mathcal{P}_T$ are not useful preconditioners, as the matrices A and S are computationally expensive to invert in general, so preconditioners of the form

$$\widehat{\mathcal{P}}_D = \begin{bmatrix} \widehat{A} & 0 \\ 0 & \widehat{S} \end{bmatrix}, \qquad \widehat{\mathcal{P}}_T = \begin{bmatrix} \widehat{A} & 0 \\ B & -\widehat{S} \end{bmatrix}$$

are sought, where $\widehat{A}$ and $\widehat{S}$ denote suitably chosen approximations of the $(1,1)$-block A and Schur complement S. The objective here is that a Krylov method will not converge in 3 or 2 iterations, but just a few more, while at the same time ensuring that the preconditioner is much cheaper to invert.

However, as highlighted above, much classical saddle point theory relies on the matrix A being invertible. If this is not the case, as we shall find for the application under consideration here, a modification to this approach must be sought.

4.1. Construction of the preconditioner. We first examine the system (3.2), and place this in the saddle point form (4.1) as follows:

$$A = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 & 0 \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) & 0 \\ 0 & 0 & \gamma_u \cdot \mathrm{Id} \end{bmatrix}, \qquad B = \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & -\beta\chi_{\partial\Omega}(\blacksquare) \end{bmatrix}.$$

Furthermore, let us decompose the blocks A and B into sub-blocks:

$$A = \begin{bmatrix} A_s & 0 \\ 0 & A_u \end{bmatrix}, \qquad B = \begin{bmatrix} B_s & B_u \end{bmatrix},$$

where

$$A_s = \begin{bmatrix} \chi_{\Omega_T}(\blacksquare) & 0 \\ 0 & \gamma_c\chi_{\Omega_T}(\blacksquare) \end{bmatrix}, \qquad B_s = \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} \end{bmatrix}, \qquad B_u = \begin{bmatrix} 0 \\ -\beta\chi_{\partial\Omega}(\blacksquare) \end{bmatrix}.$$

In this paper, A_u corresponds to the finite element discretization of the following operators:

$$A_u \leftarrow \begin{cases} \gamma_u \cdot \mathrm{Id} & \text{without control constraints,} \\ \gamma_u \cdot \mathrm{Id} + \frac{1}{\delta}G_\Lambda & \text{with control constraints,} \end{cases}$$

that is to say the block A_u is altered if we instead consider the matrix (3.3) incorporating control constraints. Note that, as the saddle point system is written, the matrix A is not invertible, as the matrix A_s is only positive semi-definite with the vast majority of eigenvalues equal to zero. The Schur complement S, as defined above, therefore does not exist. However, the matrices A_u, B_s and B_s^T are invertible, and we wish to make use of inexact solves for some of these blocks within our preconditioner.

We hence consider a suitable re-ordering of the matrices (3.2) and (3.3) to enable us to utilize a variant of the classical saddle point theory outlined above.

In particular, we observe that the matrix under consideration may be factorized as follows:

$$\begin{bmatrix} A_s & 0 & B_s^\top \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix} = \begin{bmatrix} I & -A_s B_s^{-1} B_u A_u^{-1} & A_s B_s^{-1} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \underbrace{\begin{bmatrix} 0 & 0 & S_\angle \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix}}_{\mathcal{P}}, \quad (4.2)$$

where identity matrices I are of appropriate dimensions. Note that as Id corresponds to the identity operator on the continuous level, this will become a finite element mass matrix in the discrete setting. We then take $\mathcal{P}$ to be the foundation of our preconditioner. We define $S_\angle$ as the '*pivoted Schur complement*' [6, Sec. 3.3]

$$S_\angle = B_s^\top + A_s B_s^{-1} B_u A_u^{-1} B_u^\top. \quad (4.3)$$

We approximate this term within our preconditioner using the '*matching strategy*' devised in [41, 43, 44], which aims to capture both terms of the Schur complement within the preconditioner. The approximation reads as follows:

$$S_\angle \approx \widehat{S}_\angle := \Big(B_s^\top + \frac{1}{\eta} A_s\Big) B_s^{-1} \Big(B_s + \eta B_u A_u^{-1} B_u^\top\Big). \quad (4.4)$$

Note that the matrix product $B_s^\top B_s^{-1} B_s$ captures the first term $B_s^\top$ of $S_\angle$, and $\big(\frac{1}{\eta} A_s\big) B_s^{-1} \big(\eta B_u A_u^{-1} B_u^\top\big)$ matches exactly the second term $A_s B_s^{-1} B_u A_u^{-1} B_u^\top$. The positive constant η is chosen to 'balance' the first and last matrix factors, $B_s^\top + \frac{1}{\eta} A_s$ and $B_s + \eta B_u A_u^{-1} B_u^\top$, within the Schur complement approximation, so that the two terms in the remainder $S_\angle - \widehat{S}_\angle$ are approximately of the same norm, or contribute in a roughly equal way to the characteristics of the matrix. Two natural choices for this constant are

$$\eta = \sqrt{\frac{\|A_s\|}{\big\|B_u A_u^{-1} B_u^\top\big\|}} \quad \text{or} \quad \eta = \sqrt{\frac{\max(\operatorname{diag}(A_s))}{\max(\operatorname{diag}(B_u A_u^{-1} B_u^\top))}}.$$

We note that the second choice is much cheaper to compute, and it is anticipated (and observed in practice) that the maximum diagonal entries of A_s and $B_u A_u^{-1} B_u^\top$ offer representative values indicating the ratio of the contributions of the two matrices. Approximately solving for the matrix $B_s + \eta B_u A_u^{-1} B_u^\top$ is made tractable by the effective approximation of a mass matrix (or a mass matrix plus a positive diagonal matrix) of the form A_u by its diagonal, see [39, Sec. 4.1] and [65].

Putting all the pieces together, we state our preconditioner

$$\widehat{\mathcal{P}} = \begin{bmatrix} 0 & 0 & \widehat{S}_\angle \\ 0 & A_u & B_u^\top \\ B_s & B_u & 0 \end{bmatrix},$$

incorporating the Schur complement approximation above. Due to the re-ordering of the saddle point system that we have undertaken, this is a suitable choice of preconditioner that should capture the characteristics of the matrix under consideration. Indeed, one may directly compute that $\mathcal{P}\widehat{\mathcal{P}}^{-1} = \operatorname{blkdiag}(S_\angle \widehat{S}_\angle^{-1}, I, I)$. Along with

(4.2), this suggests that $\widehat{\mathcal{P}}$ is a good candidate for a preconditioner, given a suitable approximation $\widehat{S}_{\angle}$ of $S_{\angle}$. It should be emphasized that the convergence of a non-symmetric solver such as GMRES [49], within which $\widehat{\mathcal{P}}$ must be applied, cannot be theoretically guaranteed through eigenvalue analysis of the preconditioned system, however for the problem under consideration we find that this preconditioner leads to both clustered eigenvalues and good convergence properties.

4.2. Application of the preconditioner. Applying the inverse of the preconditioner, $\widehat{\mathcal{P}}^{-1}$, as is necessary within an iterative method, therefore requires three main operations:

1. Applying B_s^{-1}: This is equivalent to solving the linearized forward problem, rather than the coupled optimization problem. In practice this is approached time-step by time-step, using an algebraic or geometric multigrid method, or another suitable scheme, to solve for the matrices arising at each point in time.
2. Applying A_u^{-1}: The matrix A_u is a block diagonal matrix, consisting of boundary mass matrices at each time-step (in the case without control constraints), or boundary mass matrices plus positive semidefinite diagonal matrices (if control constraints are present). In either case, these matrices may be well approximated using Chebyshev semi-iteration [11, 12, 66], or even using a simple diagonal approximation of a mass matrix [65].
3. Applying $\widehat{S}_{\angle}^{-1}$: Applying the approximation (4.4) involves a multiplication operation involving B_s, and (approximate) solves for each of $B_s^{\top} + \frac{1}{\eta}A_s$ and $B_s + \eta B_u A_u^{-1} B_u^{\top}$ which may again be approached at each time-step in turn using multigrid or another appropriate method.

4.3. Uzawa approximation. In practice, we make a further modification to the preconditioner $\widehat{\mathcal{P}}$ in order to ensure it is easier to work with on a computer. In more detail, the term B_s in the bottom-left of $\widehat{\mathcal{P}}$, and the terms $B_s^{\top} + \frac{1}{\eta}A_s$ and $B_s + \eta B_u A_u^{-1} B_u^{\top}$ within $\widehat{S}_{\angle}$, contain 2×2 block systems which we would like to replace with more convenient approximations so that we are only required to (approximately) invert one block at a time.

To facilitate this, we replace 2×2 block matrices by an inexact Uzawa approximation, with block triangular splitting matrices, where appropriate. This leads to our final choice of preconditioner:

$$\widetilde{\mathcal{P}} = \begin{bmatrix} 0 & 0 & \left(B_s^{\top} + \frac{1}{\eta}A_s\right)_{\text{Uzawa}} (B_s)_{\text{Uzawa}}^{-1} \left(B_s + \eta B_u A_u^{-1} B_u^{\top}\right)_{\text{Uzawa}} \\ 0 & A_u & B_u^{\top} \\ (B_s)_{\text{Uzawa}} & B_u & 0 \end{bmatrix},$$

where $(\cdot)_{\text{Uzawa}}$ denotes the Uzawa approximation of the corresponding matrix. For ease of reproducibility for the reader, we state the splitting matrices below:

$$\begin{aligned} (B_s)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ 0 & \mathcal{L}_{qc} \end{bmatrix}, \\ \left(B_s^{\top} + \frac{1}{\eta}A_s\right)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{zp} + \frac{1}{\eta}\chi_{\Omega_T}(\blacksquare) & 0 \\ \mathcal{L}_{cp} & \mathcal{L}_{cq} + \frac{\gamma_c}{\eta}\chi_{\Omega_T}(\blacksquare) \end{bmatrix}, \\ \left(B_s + \eta B_u A_u^{-1} B_u^{\top}\right)_{\text{Uzawa}} &\to \begin{bmatrix} \mathcal{L}_{pz} & \mathcal{L}_{pc} \\ 0 & \mathcal{L}_{qc} + \eta\beta^2 \chi_{\partial\Omega}(\blacksquare) A_u^{-1} \chi_{\partial\Omega}(\blacksquare)^{\top} \end{bmatrix}. \end{aligned}$$

Now the linear systems with diagonal blocks ($\mathcal{L}_{zp}$, $\mathcal{L}_{qc}$, and so on) can be solved directly. Note that it is also possible to annihilate another off-diagonal block instead within the Uzawa approximation. However, we have found that the approximations listed above yield fast convergence in our numerical experiments. Ideally, one can carry out a number of Uzawa iterations to approximate the inversion of $\widehat{S}_{\angle}$ with a target accuracy. In practice, however, we have found that just a single application of $\widetilde{\mathcal{P}}$ (thus called "approximation") is sufficient for rapid convergence of GMRES when applied to the systems (3.2) or (3.3).

5. Numerical experiments with control constraints. In this section we benchmark the preconditioned Newton method. For our test problem, the initial distribution of bacterial cells is chosen as a sum of m_0 independent Gaussian peaks,

$$z_0(x,y) = \sum_{i=1}^{m_0} \exp\left(-2560 \cdot \left[(x-x_i)^2 + (y-y_i)^2\right]\right), \tag{5.1}$$

where the centers $\{x_i, y_i\}$ are chosen randomly on $[0,1]^2$. This square domain is partitioned into uniform squares of size $h \times h$ each, where $h = 1/(n-1)$, and the solution components z, c, u, p, q are approximated in a product basis of n piecewise linear finite elements in each variable, see (6.1)–(6.2) in the next section with $n_1 = n_2 = n$. Moreover, we discretize the time derivative with an implicit Euler scheme with $n_3 = n$ time steps. This yields n^3 unknowns in the vectors of coefficients of each solution component, totalling $5n^3$ degrees of freedom for the entire discrete solution. In principle, we could use any grid and finite element basis. However, in order to compare CPU times with those of the low-rank decomposition approach introduced in the next section, we continue with the Cartesian ansatz.

The desired distribution at the final time $T = 1$ is linear,

$$\widehat{z}(x,y) = \langle z_0 \rangle \cdot (x+y), \tag{5.2}$$

normalized by the initial mass,

$$\langle z_0 \rangle = \int_{[0,1]^2} z_0(x,y)\ \mathrm{d}x\mathrm{d}y,$$

since the model conserves the normalization of z. Both initial and target concentrations c are zero. The experiments were run in MATLAB R2017b on one core of a 2.4GHz Intel Xeon E5-2640 CPU.

In this section, we set $m_0 = 50$ and the control constraints $u_- = 0$ and $u_+ = 0.2$, in accordance with [46]. The default regularization parameters are set to $\gamma_u = 10^{-3}$ and $\gamma_c = 0.5$. The stopping tolerance for the Newton iteration is set to $\varepsilon = 10^{-4}$. The Newton method is stopped at the jth iteration when the relative 2-norm of the increment of all coefficient vectors becomes smaller than the tolerance,

$$\max\left(\frac{\|\mathbf{z}_j - \mathbf{z}_{j-1}\|_2}{\|\mathbf{z}_j\|_2}, \frac{\|\mathbf{c}_j - \mathbf{c}_{j-1}\|_2}{\|\mathbf{c}_j\|_2}, \frac{\|\mathbf{u}_j - \mathbf{u}_{j-1}\|_2}{\|\mathbf{u}_j\|_2}, \frac{\|\mathbf{p}_j - \mathbf{p}_{j-1}\|_2}{\|\mathbf{p}_j\|_2}, \frac{\|\mathbf{q}_j - \mathbf{q}_{j-1}\|_2}{\|\mathbf{q}_j\|_2}\right) < \varepsilon.$$

Moreover, we decrease the Moreau–Yosida regularization parameter δ geometrically from 10^{-1} to 10^{-4} as the iteration converges. This gives more robust behavior of the Newton method.

The computational time is shown in Fig. 5.1 (left). We solve the linear systems arising in the Uzawa preconditioner described in Section 4.3 using direct elimination,

FIG. 5.1. *Left: CPU time of the Newton solver for the constrained control. Right:* $u(\mathbf{x}, T/2)$.

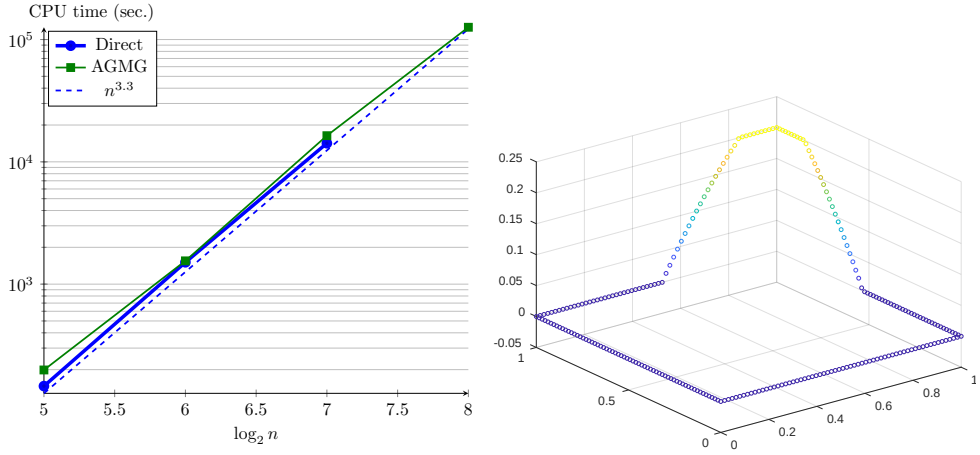
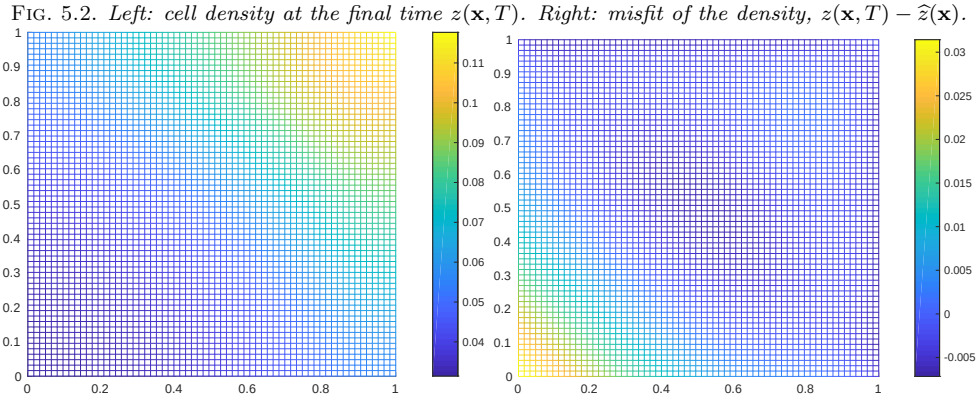


FIG. 5.2. *Left: cell density at the final time* $z(\mathbf{x}, T)$. *Right: misfit of the density,* $z(\mathbf{x}, T) - \widehat{z}(\mathbf{x})$.



as well as the AGMG (AGgregation-based algebraic MultiGrid) V-cycle iteration [33, 34, 35, 36] version 3.1.2 (recent version available from `http://agmg.eu/`) with a tight stopping threshold of 10^{-12}. We see that the CPU times grow slightly faster than cubically with respect to the uniform grid refinement, which is expected for a three-dimensional (2D space + time) problem. It should be noted that the full solution procedure consumes a considerable amount of memory. For example, for $n = 256$ each solution component is discretized on nearly 17 millions degrees of freedom, but 10 finite element matrices, each containing up to 9 nonzero entries per row, occupy about 25Gb of memory in total. The linear solvers increase this amount further, to 33Gb with AGMG and to over 64Gb with the direct solver, with the latter quantity exceeding the capacity of the machine used.

On the other hand, the number of Newton iterations is quite stable with respect to the grid size, ranging from 11 to 14 depending on a particular distribution of the random initial guess.

The transient control signal is shown in Fig. 5.1 (right). We notice that it is accurately confined within the prescribed constraints. However, this leads to a rather large misfit in the target cell density (Fig. 5.2). While the density follows the linear distribution $\widehat{z}$ correctly in the top right corner of the domain, in the left bottom

corner we see an excessive density of bacteria. This shows that controlling only the chemoattractant might be insufficient for forcing the bacteria to leave a particular area.

Lastly in this section, we investigate the performance of the preconditioner proposed in Section 4 against variation of parameters. In Table 5.1, we show the average numbers of GMRES iterations per Newton step for different grid sizes n and regularization parameters γ_u, γ_c, using direct solves for the Uzawa preconditioner. We vary only one parameter at a time, while the other two are kept fixed to their default values, $n = 64$, $\gamma_u = 10^{-3}$, and $\gamma_c = 0.5$. The number of iterations grows slightly as the control regularization parameter γ_u is decreased, which is expected for a boundary control problem. On the other hand, the preconditioner is reasonably robust with respect to the other parameters, in particular the grid size.

TABLE 5.1
Average number of GMRES iterations per Newton step.

n	its
32	21.37
64	27.46
128	27.86

γ_u	its
10^{0}	6.00
10^{-1}	9.00
10^{-2}	15.28
10^{-3}	27.46
10^{-4}	46.84
10^{-5}	69.21

γ_c	its
0.5	27.46
10^{-1}	30.55
10^{-2}	32.11
10^{-3}	31.77
10^{-4}	32.67
10^{-5}	31.57

6. Low-rank tensor decompositions and algorithms. The optimality system (3.2) can result in a huge-scale matrix system, for many spatial degrees of freedom and time steps. One way to reduce the associated computational burden is to seek an approximate solution in a low-parametric representation. In this paper we apply separation of variables, and in particular the Tensor Train (TT) decomposition [37]. In this section, we introduce the TT decomposition and the algorithm for an efficient TT-structured solution of the optimality equations. Although the TT approximation can have difficulties with the indicator function of the active set of control constraints (see Remark 5 below), for the problem without box constraints it yields a very efficient solver. So in this section we assume an unconstrained control setting.

6.1. Tensor product discretization and indexing. We assume that the solution functions can be discretized on a structured grid, e.g., the cell concentration $z(\mathbf{x}, t)$ with a d-dimensional spatial variable $\mathbf{x} = (x_1, \dots, x_d)$ can be approximated by

$$z(\mathbf{x}, t) \approx \sum_{j_1,\dots,j_d,j_{d+1}=1}^{n_1,\dots,n_d,n_{d+1}} \mathbf{z}(j_1, \dots, j_d, j_{d+1}) \phi_{j_1,\dots,j_d}(\mathbf{x}) \psi_{j_{d+1}}(t), \tag{6.1}$$

where $\{\phi_{j_1,\dots,j_d}(\mathbf{x})\}$ is a set of spatial basis functions as introduced in Section 3, which we now assume to be indexed by d independent variables. In particular, we consider a square domain $\mathbf{x} \in [0, 1]^d$ and the piecewise polylinear basis functions

$$\phi_{j_1,\dots,j_d}(\mathbf{x}) = \varphi_{j_1}(x_1) \cdots \varphi_{j_d}(x_d). \tag{6.2}$$

In turn, $\{\psi_{j_{d+1}}(t)\}$ is a set of nodal interpolation functions in time, associated with the uniform time grid $\{t_{j_{d+1}}\}$, with $t_{j_{d+1}} = \tau \cdot j_{d+1}$, $j_{d+1} = 1, \dots, n_{d+1}$, and $\tau = T/n_{d+1}$. We can see that the discrete coefficients of z can be collected into a $(d+1)$-dimensional

tensor. If we restrict ourselves to the isotropic discretization with $n_1 = \cdots = n_{d+1} = n > 1$, we arrive at n^{d+1} entries in the tensor $\mathbf{z}$. The computational complexity of solving (3.2) is usually much higher. This explains the sometimes relatively high computing times in the previous section.

Separation of the discrete variables $j_1, \ldots, j_{d+1}$ can compress the tensor data from the exponential $\mathcal{O}(n^{d+1})$ to a linear volume $\mathcal{O}(dn)$. Yet we can aim for a higher compression ratio. Assuming that n_k is factorizable into a set of divisors $n_{k,1} \cdots n_{k,L_k} = n_k$, $n_{k,p} > 1$, $p = 1, \ldots, L_k$ (we aim for minimal divisors, e.g., we only use $n_{k,p} = 2$ and 3 in the numerical experiments), we can also factorize the index j_k into the corresponding digits,

$$j_k = 1 + \sum_{\ell=1}^{L_k} (i_{\overline{\ell,k}} - 1) \prod_{p=1}^{\ell-1} n_{k,p}, \qquad k = 1, \ldots, d+1.$$

Here $\overline{\ell, k} = \ell + \sum_{p=1}^{k-1} L_p =: m$ is the lexicographic combination of the individual indices ℓ, k, such that $m = 1, \ldots, L = \sum_{k=1}^{d+1} L_k$. Now the tensor $\mathbf{z}$ can be enumerated by the elementary digits i_m. Instead of considering $\mathbf{z}$ as a $(d+1)$-dimensional tensor, we treat it as a L-dimensional tensor with elements $\mathbf{z}(i_1, \ldots, i_L)$, and therefore we will separate now the *virtual* indices i_m [59].

6.2. Tensor Train decomposition. As a particular separated approximation, we choose the *Tensor Train* (TT) decomposition [37], which is also known as the Matrix Product States [45, 55] in physics:

$$\mathbf{z}(i_1, \ldots, i_L) \approx \sum_{s_1=1}^{r_1} \cdots \sum_{s_{L-1}=1}^{r_{L-1}} z_{s_1}^{(1)}(i_1) z_{s_1,s_2}^{(2)}(i_2) \cdots z_{s_{L-1}}^{(L)}(i_L). \tag{6.3}$$

The factors $z^{(m)}$ on the right hand side are called *TT blocks*, and the ranges $r_1, \ldots, r_{L-1}$ of the auxiliary summation indices are called *TT ranks*. Notice that the TT blocks are at most 3-dimensional tensors, of sizes $r_{m-1} \times \bar{n}_m \times r_m$ (for uniformity, we can let $r_0 = r_L = 1$).

Potentially, we can represent any finite dimensional tensor exactly through (6.3) by choosing large enough TT ranks. For reasons of numerical efficiency, we will of course aim for a (sub-)optimal approximation with r_m being as small as possible, and most importantly much smaller than the original tensor size $n_1 \cdots n_{d+1}$. The storage needed for the right hand side of (6.3) is of $\mathcal{O}(L \bar{n}_m r_m^2)$, where $\bar{n}_m := n_{k,\ell}$ is also chosen to be much smaller than the original grid sizes n_k. For example, if we restrict the grid sizes to be powers of two, $n_k = 2^{L_k}$, the range of each index i_m in (6.3) becomes just $\{1, 2\}$, whereas L, and hence the storage complexity of the TT format, becomes *logarithmic* in the original tensor size, $L = \log_2(n_1 \cdots n_{d+1})$. Due to the minimal non-trivial index range in this case, the TT decomposition (6.3) with $i_m \in \{1, 2\}$ was called the *Quantized* TT (QTT) decomposition [24]. It was then proved that many examples of vectors [24] and matrices [21, 22], arising from the discretization of functions and differential operators, allow low-rank QTT decompositions.

Abstracting from the original problem dimensions, we can consider only two data representations: a tensor with the smallest possible ranges $\mathbf{z}(i_1, \ldots, i_L)$, and a vector of the same data entries:

$$\mathbf{z}(j) = \mathbf{z}(i_1, \ldots, i_L), \quad \text{where} \quad j = 1, \ldots, (\bar{n}_1 \cdots \bar{n}_L). \tag{6.4}$$

We need the vector notation for setting the Gauss–Newton equations (3.2) on tensors consistently. Boldface letters (e.g., $\mathbf{z}$) from now on will denote vectors. We can use the Kronecker product ($\otimes$) to rewrite (6.3) in an equivalent vector form,

$$\mathbf{z} = \sum_{s_1,\ldots,s_{L-1}=1}^{r_1,\ldots,r_{L-1}} z^{(1)}_{s_1} \otimes z^{(2)}_{s_1,s_2} \otimes \cdots \otimes z^{(L)}_{s_{L-1}}. \tag{6.5}$$

Of course, we shall never actually compute the Kronecker products in the expansion above, but only store and manipulate individual TT blocks on the right hand side.

For example, the matrix-vector product $\mathbf{y} = A\mathbf{z}$ with $\mathbf{z}$ given in (6.3) can be computed efficiently if we can also represent the matrix by a TT decomposition,

$$A = \sum_{s_1,\ldots,s_{L-1}=1}^{R_1,\ldots,R_{L-1}} A^{(1)}_{s_1} \otimes A^{(2)}_{s_1,s_2} \otimes \cdots \otimes A^{(L)}_{s_{L-1}}. \tag{6.6}$$

For example if the matrix is diagonal, and the vector of the diagonal values can be represented by a TT decomposition (6.3), the matrix can be written as in (6.6), with the same TT ranks. There are less trivial matrices, arising for example in finite element computations, that admit TT decompositions with modest ranks R_m [21, 22]. Now the result $\mathbf{y} = A\mathbf{z}$ can be also written in the TT format and computed block by block. Moreover, a TT decomposition with excessive TT ranks can be efficiently approximated up to a desired accuracy by a decomposition with sub-optimal ranks using QR and singular value decomposition (SVD) factorizations [37], without ever constructing full large tensors.

6.3. Alternating Linear Scheme iteration. Considering a linear equation $A\mathbf{z} = \mathbf{y}$ where A is assumed to be given in the TT format (6.6), $\mathbf{y}$ is assumed to be given in a counterpart of (6.5), and $\mathbf{z}$ is assumed to be approximable by (6.5), we can construct an algorithm for computing directly the TT blocks of an approximation of $\mathbf{z}$. The idea is to plug the decomposed form of $\mathbf{z}$ into the original equation and solve the corresponding overdetermined system for the elements of only one TT block of $\mathbf{z}$ at a time. A crucial ingredient for the efficient solution of this constrained system is the *linearity* of the TT format. For each $m = 1, \ldots, L$, we can construct the so-called *frame* matrix, where the TT block $z^{(m)}$ in (6.5) is replaced by the identity matrix,

$$Z_m = \left(\sum_{s_1,\ldots,s_{m-2}} z^{(1)}_{s_1} \otimes \cdots \otimes z^{(m-1)}_{s_{m-2}} \right) \otimes I \otimes \left(\sum_{s_{m+1},\ldots,s_{L-1}} z^{(m+1)}_{s_{m+1}} \otimes \cdots \otimes z^{(L)}_{s_{L-1}} \right). \tag{6.7}$$

Stretching all elements of $z^{(m)}$ into a vector $\mathbf{z}^{(m)} \in \mathbb{R}^{r_{m-1}\bar{n}_m r_m}$, we can observe that

$$\mathbf{z} = Z_m \mathbf{z}^{(m)}, \tag{6.8}$$

i.e., the frame matrix realizes a linear map from the elements of $z^{(m)}$ to the elements of the whole solution vector. This motivates an iterative algorithm [18], which was called the Alternating Linear Scheme (ALS):

1: **for** iter $= 0, 1, \ldots$ until convergence **do**
2: **for** $m = 1, 2, \ldots, d, d-1, \ldots, 1$ **do**
3: Plug the solution in the form (6.8) into the original problem $A\mathbf{z} = \mathbf{y}$.
4: Solve the resulting overdetermined problem $(AZ_m)\mathbf{z}^{(m)} = \mathbf{y}$ on $\mathbf{z}^{(m)}$.

5: Put new $z^{(m)}$ back into (6.5) and prepare frame matrix (6.7) for $m \pm 1$.
6: **end for**
7: **end for**

Starting from a low-rank initial guess of the form (6.9), this algorithm seeks the solution in a low-rank TT format by sweeping through the different TT blocks. However, there might be different ways to resolve the overdetermined problem in Line 4. The simplest option, which can be justified for a symmetric positive definite A, is to project the problem onto the same frame matrix and solve a small linear system $(Z_m^\top A Z_m)\mathbf{z}^{(m)} = Z_m^\top \mathbf{y}$ in each step. However, the matrix in (3.2) is indefinite, since we seek a saddle point of the Lagrangian. Therefore, in the next section we develop a different version of the ALS algorithm.

6.4. Block TT representation and ALS for solving (3.2)**.** Another property of the optimality system (3.2) is that we need to approximate several solution components in addition to the cell concentration $z(\mathbf{x}, t)$. Since the components z, c, p, q are defined on the same domain, with u defined on its boundary, we can discretize them using the same basis. The tensors of discrete values for z, c, p, q therefore have the same sizes. The structure of the problem (3.2) suggests that we approximate them in a shared TT decomposition, the so-called *block* TT format [8]. We denote the aggregated solution

$$\mathbf{y}^\top = \begin{bmatrix} \mathbf{z}^\top & \mathbf{c}^\top & \mathbf{p}^\top & \mathbf{q}^\top & \mathbf{u}^\top \end{bmatrix},$$

enumerating the components via $\mathbf{y}_j$, $j = 1, \ldots, 5$. Now we decompose $\mathbf{y}$ into a TT format with all the same TT blocks except the m-th block for some $m = 1, \ldots, L$, which actually carries the enumerator of the components,

$$\mathbf{y}_j = \sum_{s_1,\ldots,s_{L-1}=1}^{r_1,\ldots,r_{L-1}} y^{(1)}_{s_1} \otimes \cdots \otimes y^{(m-1)}_{s_{\ell-2},s_{\ell-1}} \otimes \widehat{y}^{(m)}_{s_{\ell-1},s_\ell}(j) \otimes y^{(m+1)}_{s_\ell,s_{\ell+1}} \otimes \cdots \otimes y^{(L)}_{s_{L-1}}. \quad (6.9)$$

Moreover, we can switch between the representations (6.9) corresponding to different m (and hence having j in different TT blocks) using the SVD [8]. For example, we can reshape $\widehat{y}^{(m)}$ into a matrix with elements

$$\widehat{Y}^{(m)}(s_{m-1}, i_m;\ j, s_m) = \widehat{y}^{(m)}_{s_{m-1},s_m}(i_m, j)$$

and compute the truncated SVD $\widehat{Y}^{(m)} \approx U\Sigma V^\top$. Now we write the left singular vectors U into the m-th TT block instead of $\widehat{y}^{(m)}$, and multiply $\Sigma V^\top$ with the $(m+1)$-th TT block,

$$y^{(m)}_{s_{m-1},s'_m}(i_m) = U(s_{m-1}, i_m;\ s'_m), \quad (6.10)$$

$$\widehat{y}^{(m+1)}_{s'_m,s_{m+1}}(i_{m+1}, j) = \sum_{s_m=1}^{r_m} \Sigma V^\top(s'_m;\ j, s_m) y^{(m+1)}_{s_m,s_{m+1}}(i_{m+1}). \quad (6.11)$$

Note that we have obtained the same representation as (6.9) with m replaced by $m+1$. This process can be continued further, or reversed, and hence the j-index can be placed into any TT block.

The mth frame matrix for the block TT decomposition (6.9) is defined in the

same way as (6.7):

$$Y_m = \left(\sum_{s_1,\dots,s_{m-2}} y^{(1)}_{s_1} \otimes \cdots \otimes y^{(m-1)}_{s_{m-2}} \right) \otimes I \otimes \left(\sum_{s_{m+1},\dots,s_{L-1}} y^{(m+1)}_{s_{m+1}} \otimes \cdots \otimes y^{(L)}_{s_{L-1}} \right). \tag{6.12}$$

However, notice that $Y_m \in \mathbb{R}^{n^{d+1} \times (r_{m-1}\bar{n}_m r_m)}$ is free from the special block $\widehat{y}^{(m)}$ which accounts for different solution components. The columns of the matrix Y_m can thus be seen as a common basis for all components. The *Block ALS* algorithm [1, 7] projects each individual submatrix in (3.2) onto Y_m, which gives a linear system with a smaller size but the same block structure as (3.2) on the elements of $\widehat{y}^{(m)}$. However, in our specific case we can reduce the system even further by noticing that the $(3,3)$-block of (3.2) is simply a diagonal matrix in the case where lumped mass matrices are considered, and therefore eliminate the control component from the equations[2]. Specifically, we deduce that $s_u = A_u^{-1}\left(\mathbf{b}_u + \beta \chi_{\partial\Omega}^\top s_q\right)$ and plug this into the fifth row. This gives us a system of 4 equations only. Moreover, instead of using the increments s_z, s_c, s_p, s_q, we can rewrite the equations for the new solution components directly:

$$\begin{bmatrix} \chi_{\Omega_T} & 0 & \mathcal{L}_{zp} & \mathcal{L}_{zq} \\ 0 & \gamma_c \chi_{\Omega_T} & \mathcal{L}_{cp} & \mathcal{L}_{cq} \\ \mathcal{L}_{pz} & \mathcal{L}_{pc} & 0 & 0 \\ \mathcal{L}_{qz} & \mathcal{L}_{qc} & 0 & -\beta^2 \chi_{\partial\Omega} A_u^{-1} \chi_{\partial\Omega}^\top \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{c} \\ \mathbf{p} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{b}}_z \\ \widetilde{\mathbf{b}}_c \\ \widetilde{\mathbf{b}}_p \\ \widetilde{\mathbf{b}}_q \end{bmatrix},$$

where $\widetilde{\mathbf{b}}$ is the correspondingly adjusted right hand side. Now we plug in the solutions in the form $\mathbf{y}_j = Y_m \widehat{y}^{(m)}(j)$ using the frame matrix from (6.12), with j now running only from 1 to 4, and project each of the previous equations onto Y_m. This gives us a *reduced* system

$$\begin{bmatrix} \widehat{\chi}_{\Omega_T} & 0 & \widehat{\mathcal{L}}_{zp} & \widehat{\mathcal{L}}_{zq} \\ 0 & \gamma_c \widehat{\chi}_{\Omega_T} & \widehat{\mathcal{L}}_{cp} & \widehat{\mathcal{L}}_{cq} \\ \widehat{\mathcal{L}}_{pz} & \widehat{\mathcal{L}}_{pc} & 0 & 0 \\ \widehat{\mathcal{L}}_{qz} & \widehat{\mathcal{L}}_{qc} & 0 & -\beta^2 \widehat{\chi}^2_{\partial\Omega} \end{bmatrix} \widehat{y}^{(m)} = \begin{bmatrix} Y_m^\top \widetilde{\mathbf{b}}_z \\ Y_m^\top \widetilde{\mathbf{b}}_c \\ Y_m^\top \widetilde{\mathbf{b}}_p \\ Y_m^\top \widetilde{\mathbf{b}}_q \end{bmatrix}, \tag{6.13}$$

with $\widehat{\chi}_{\Omega_T} = Y_m^\top \chi_{\Omega_T} Y_m$, $\widehat{\mathcal{L}}_{**} = Y_m^\top \mathcal{L}_{**} Y_m$ (where "$**$" stands for "zp", "zq", "cp", and so on), and $\widehat{\chi}^2_{\partial\Omega} = Y_m^\top \chi_{\partial\Omega} A_u^{-1} \chi_{\partial\Omega}^\top Y_m$ the projected square matrices. Each submatrix is of size $r_{m-1}\bar{n}_m r_m \times r_{m-1}\bar{n}_m r_m$ (with $\bar{n}_m$ being small, e.g., 2), and hence (6.13) is easy to solve. Moreover, the singular value decomposition in (6.10)–(6.11) maintains the orthogonality of the frame matrices Y_m automatically in the course of alternating iterations, provided that the initial guess is given with this property. This makes the projected submatrices well conditioned if the original matrices were so, which eventually makes the entire matrix in (6.13) invertible. We highlight that the preconditioner developed in Section 4 can also be used for solving the system (6.13).

6.5. Construction of matrices in the TT format. In the course of the Newton iteration, we need to reconstruct TT representations of the matrices in (3.2) using the new solution, in order to assemble the reduced matrices in (6.13) efficiently. Assume that we need to construct an abstract bilinear form of a nonlinear transformation

[2]Our derivation is of course valid for any invertible matrix A_u, however we wish to exploit the simplicity of the matrix structure within our solver. When consistent mass matrices are applied, we can well approximate these by their diagonals within a preconditioner, see [65].

f of the solution,

$$\mathcal{L}_{\mathcal{B}} = \int f(z, c, p, q) \nabla^p \phi_i \cdot \nabla^q \phi_j \, \mathrm{d}\mathbf{x}, \tag{6.14}$$

where $p, q \in \{0, 1\}$ are the differentiation orders, and ϕ_i, ϕ_j are the basis functions. Instead of the exact functions z, c, p, q, we work with the tensors of their values, $\mathbf{z}, \mathbf{c}, \mathbf{p}, \mathbf{q}$. The corresponding values of f can be also collected into a tensor $\mathbf{f}$ of the same size, and the original function can be approximated in the same basis, i.e.,

$$f(z(\mathbf{x}), c(\mathbf{x}), p(\mathbf{x}), q(\mathbf{x})) \approx \sum_{i_1, \ldots, i_d} \mathbf{f}(i_1, \ldots, i_d) \phi_{i_1, \ldots, i_d}(\mathbf{x}).$$

Now the computation of (6.14) involves computing analytical triple products

$$\mathcal{H}(i, j, k) = \int \phi_k \nabla^p \phi_i \cdot \nabla^q \phi_j d\mathbf{x}, \qquad i, j, k = 1, \ldots, (n_1 \cdots n_d), \tag{6.15}$$

and summing them up with the values of $\mathbf{f}$,

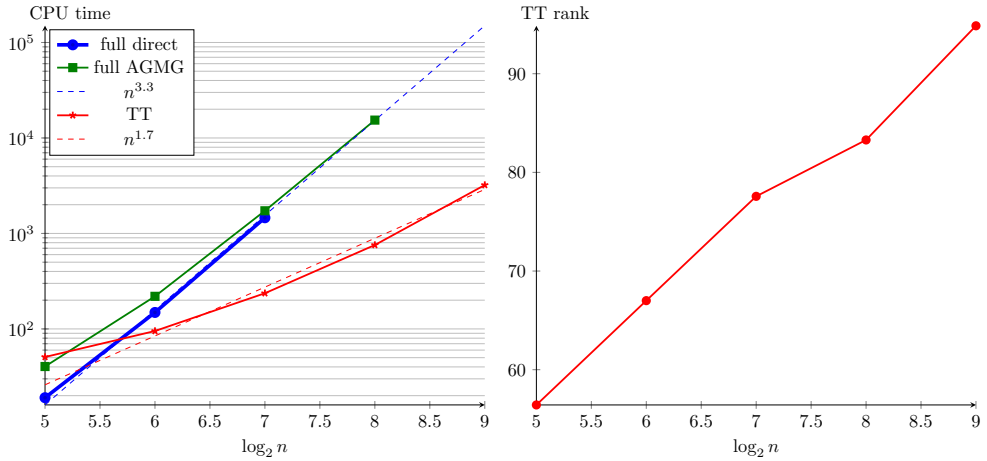
$$\mathcal{L}_{\mathcal{B}}(i, j) = \sum_{k=1}^{n_1 \cdots n_d} \mathcal{H}(i, j, k) \mathbf{f}(k). \tag{6.16}$$

Notice that we assume the basis functions can be enumerated by d independent indices, i.e., i is equivalent to $(i_1, \ldots, i_L)$ through (6.4), and similarly for j and k. The triple elements (6.15) therefore admit a TT decomposition (or even a single Kronecker-product term) similar to (6.6). Now, if the $\mathbf{f}$ tensor can also be approximated in the TT format (6.3), the bilinear form (6.14) can be represented in this format, with the TT ranks proportional (or equal) to those of $\mathbf{f}$. Moreover, the sum in (6.16) factorizes into individual sums over $k_1, \ldots, k_L$, which can be implemented efficiently block by block.

It remains to compute a TT approximation of $\mathbf{f}$. From the previous Newton iteration we are given the TT representation (6.9) for $\mathbf{z}, \mathbf{c}, \mathbf{p}, \mathbf{q}$. Hence we can rapidly evaluate any element of the solution components, and afterwards the corresponding value of f. In order to construct a TT approximation to $\mathbf{f}$ using only a few evaluations of f, we use the TT-Cross algorithm [38]. This is similar to the Alternating Linear Scheme outlined above, except that at each step it draws $r_{m-1} r_m$ fibers of the tensor values in the m-th direction in order to populate the m-th TT block and prepare the optimized fibers for the next step. In total it evaluates $\mathcal{O}(Lr^2)$ elements of the tensor, which is feasible under our assumption of small TT ranks. More robust and rank adaptive generalizations of this algorithm have followed [31, 51, 52].

REMARK 5. *Forming the diagonal of the indicator matrix G_λ in* (3.3) *seems also to be a task for the TT-Cross algorithm. However, it is likely to perform poorly in this setting, for two reasons. Firstly, if the discontinuity in a function, e.g.,* $\max\{0, u - u_+\}$*, is not aligned to coordinate axes, the corresponding TT approximation requires very large TT ranks. This can be seen already in a two-dimensional case: a triangular matrix with all ones in one of the triangles is full-rank. Secondly, the TT-Cross algorithm is likely to overlook the part of the active set which is not covered by the initial (e.g., random) set of samples. In order to adapt the sampling fibers, the cross methods require a low discrepancy between adjacent tensor elements, which is not the case for G_λ. For this reason, we apply the TT approach only to the case of the unconstrained control.*

FIG. 7.1. *CPU time (sec.) (left) and TT ranks (right) for different grid sizes n, $m_0 = 3$ initial peaks, accuracy threshold $\varepsilon = 10^{-4}$.*



7. Numerical experiments with the low-rank approximations. In this section, we benchmark the TT algorithm and compare it to the solver with the full vector representation. We use the same finite element ansatz of the Cartesian product of the piecewise linear functions. The initial distribution of bacterial cells z_0 and the desired state $\widehat{z}$ are chosen as in (5.1) and (5.2), with initial and target concentrations for the chemoattractant c set to zero. In this section the model is solved with an unconstrained control u, and final time $T = 1$. For the TT computations we use the TT-Toolbox implementation (see `https://github.com/oseledets/TT-Toolbox`).

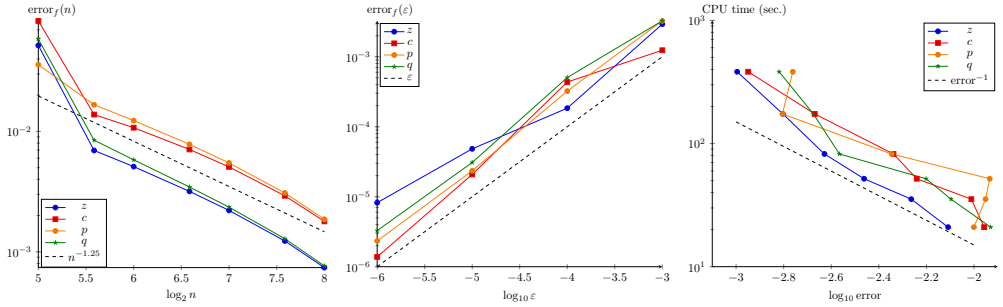
7.1. Benchmarking of full and low-rank solvers. First, we compare CPU times of the original scheme that stores full vectors with those of the approximate TT solver (see Fig. 7.1). We fix $m_0 = 3$ randomly positioned Gaussian peaks in the initial distribution z_0. Since the particular ranks and numbers of iterations depend on the choice of z_0, we average the results over 8 realizations of z_0, for each value of n. Similarly to the constrained control case, the cost of the full-format solvers grows slightly faster than cubically. On the other hand, the TT solver can proceed to much finer grids with lower time and memory footprint.

7.2. Discretization and TT approximation errors. In order to justify the use of very fine grids (up to $n = 512$), let us estimate the discretization errors. In Fig. 7.2 (left), we vary the numbers of grid points n in each direction and plot relative L^2-norm differences between the finite element approximations to the solution at a grid with n points and a grid with 512 points,

$$\text{error}_f(n) = \frac{\|f_n(T/2) - f_{512}(T/2)\|_{L^2}}{\|f_{512}(T/2)\|_{L^2}},$$

where $f_n(T/2)$ is the snapshot at $t = T/2$ of the solution component $f \in \{z, c, p, q\}$, and the L^2 norm is computed by interpolating the solution from the grid with n points to the grid with 512 points via the bilinear finite elements, and calculating the vector M-norm, where M is the mass matrix of the finite elements at the grid with 512 points. The number of initial peaks $m_0 = 3$ and their positions are fixed in these experiments.

FIG. 7.2. *Left: discretization errors for different grid sizes n and $\varepsilon = 10^{-6}$. Middle: approximation errors for different thresholds ε and $n = 64$. Right: costs for different total errors.*



We see that the error decays nearly linearly with respect to n, as expected from the implicit Euler scheme, for all quantities. The least squares log-linear fit gives a rate $\text{error}_f(n) \approx n^{-1.25}$. Since this decay is rather slow, at least 256 points in each direction are necessary to achieve an accuracy of 0.2% in the chemoattractant concentration. The seemingly faster rate of convergence near $n = 256$ is due to the reference solution computed at the grid with 512 points, which still contains a noticeable error. These points are excluded from the estimation of the rate.

The truncated singular value decomposition in the TT algorithm tries to introduce the same average amount of error to all solution components. However, the relative error in each component may differ from ε, depending on the norm scale and other factors of the algorithm, such as the local system solver. In Fig. 7.2 (middle) we investigate the relative error in all components $f \in \{z, c, p, q\}$,

$$\text{error}_f(\varepsilon) = \frac{\|f_\varepsilon - f_{10^{-8}}\|_F}{\|f_{10^{-8}}\|_F},$$

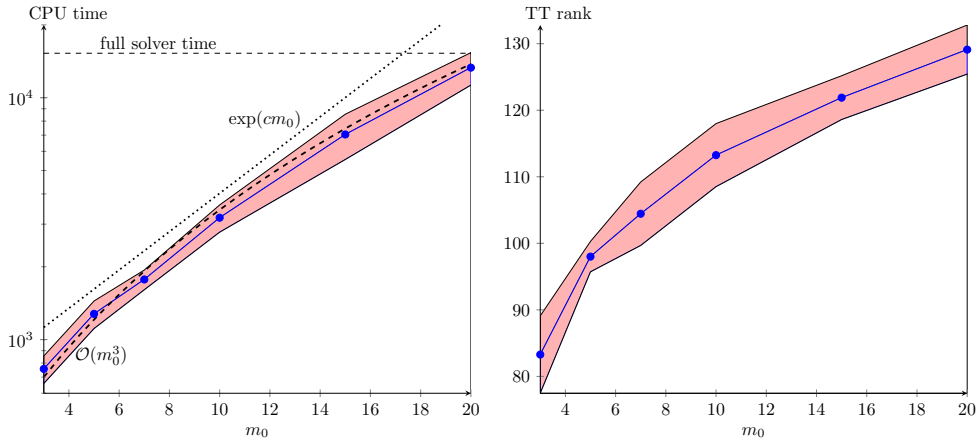
where f_ε is the solution vector computed with the TT approximation threshold ε. We see that, on average, the errors decay linearly with ε: $\text{error}_f(\varepsilon) \approx 3.2 \cdot \varepsilon$, as expected.

From the modelling point of view, one is usually interested in achieving a target total error in the numerical solution, which consists of the discretization error and the TT approximation error in our case. In this section we study how the total cost of the TT scheme depends on this total error. We vary the number of grid points n from 48 to 256, estimate the corresponding discretization error through $\text{error}_f(n) \approx n^{-1.25}$, and set the TT truncation threshold such that the errors are equal, $\text{error}_f(\varepsilon) = \text{error}_f(n)$, and hence $\varepsilon = n^{-1.25}/3.2$. This should yield the total error in the order of $2 \cdot \text{error}_f(n)$. In Fig. 7.1 (right) we plot the CPU times with respect to the total error. We see that the cost grows inversely proportional to the error, which is governed by the univariate grid size. This is a significant improvement compared to the uncompressed scheme (cf. Fig. 5.1), where the cost depends exponentially on the dimension.

7.3. Number of peaks in the initial distribution. Since the initial distribution of cells (5.1) consists of several randomly located Gaussian peaks, the particular positions of the peaks may influence the performance of the methods. In Fig. 7.3 we investigate CPU times and TT ranks in the TT solver versus the number of peaks m_0 and their positions. The plots show means plus minus standard deviations of the times and ranks with respect to the randomization of peak locations.

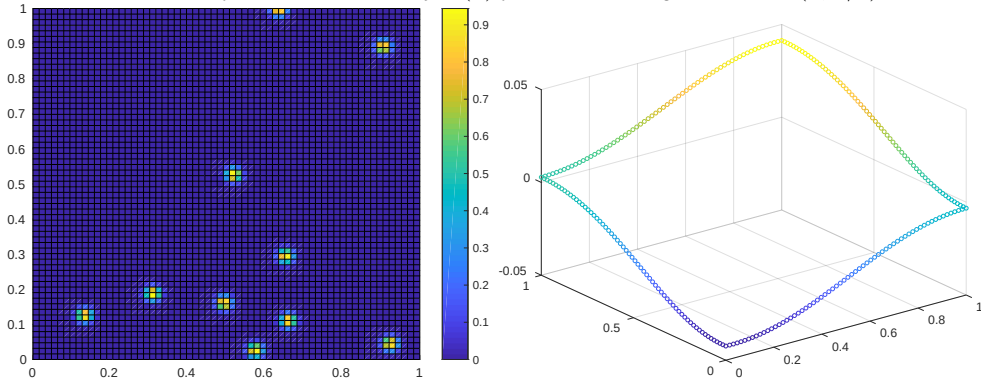
As expected, the complexity grows with the number of peaks, and for $m_0 \sim 20$ this approaches the estimated time of the full solver (should one have a sufficient

FIG. 7.3. *CPU time (sec.) (left) and TT ranks (right) for different numbers m_0 of initial peaks, accuracy threshold $\varepsilon = 10^{-4}$, $n = 256$.*



amount of memory to run the latter). For a smaller number of peaks the TT solver is more efficient. Moreover, a small relative dispersion shows that it is quite insensitive to the particular realization of the initial distribution.

FIG. 7.4. *Left: initial cell density $z_0(\mathbf{x})$ for $m_0 = 10$. Right: control $u(\mathbf{x}, T/2)$.*

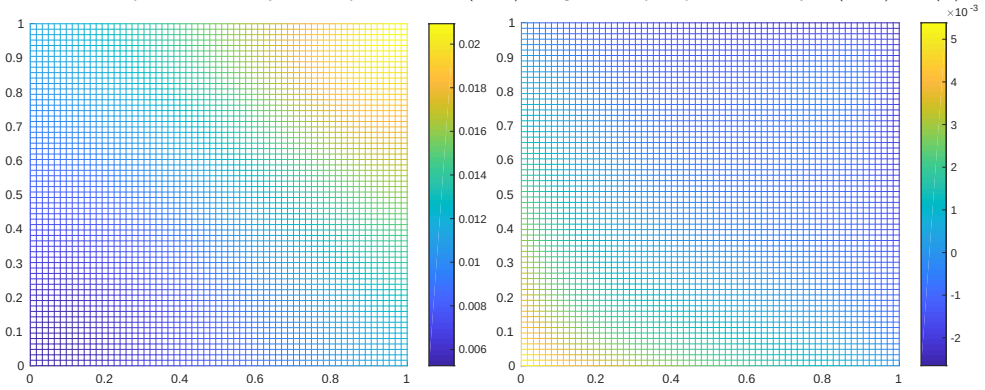


The initial cell density for $m_0 = 10$ peaks and the transient control signal are shown in Fig. 7.4 (left and right, respectively), while the final density and the misfit are shown in Fig. 7.5. The unconstrained control takes negative values in the left bottom corner of the domain. However, this gives a more accurate fit of the cell density to the desired distribution than the constrained control.

REMARK 6. *An unconstrained control can potentially drive the model into an unphysical regime, e.g., making the concentrations negative. For this example we have verified that both concentrations remain positive, and hence the Keller–Segel model remains valid.*

8. Concluding remarks. We have developed a preconditioned Gauss–Newton method for solving optimal control problems in chemotaxis, making use of an effective saddle point type preconditioner coupled with a suitable approximation of the pivoted Schur complement. This enables us to solve potentially huge-scale matrix systems,

FIG. 7.5. *Left: cell density at the final time* $z(\mathbf{x}, T)$. *Right: misfit of the density,* $z(\mathbf{x}, T) - \widehat{z}(\mathbf{x})$.



both without and with additional box constraints imposed on the control variable. Numerical results indicate considerable robustness with respect to the matrix dimension, as well as the parameters involved in the problem set-up.

Moreover, we have shown that the problem without box constraints is amenable to a faster solution using the low-rank tensor approximations of all vectors and matrices arising in the discretization. The nonlinearity of the problem can easily be tackled via cross approximation methods, provided that the functions are smooth. The low-rank decompositions are not very suitable for discontinuous functions, such as an indicator of an active set, arising in the problem of finding a constrained control or state. However, in the unconstrained case the low-rank algorithms are much faster and need much less memory than the straightforward solution of the Gauss–Newton equations. The tensor decomposition approach requires a structured grid without local refinements, which might produce overly many grid points at first glance. However, these points are never treated explicitly. Instead, only the elements of the low-rank factors are actually computed, which compensates for the fine original grid, as the low-rank decompositions usually give a more significant cost reduction compared to the grid refinement. Depending on the "complexity" of the transient solution (and hence the tensor ranks), we can achieve a speedup of more than an order of magnitude.

The importance of the box constraints depends on the particular model. For example, if we can only control the inflow of the chemoattractant, it is reasonable to request a nonnegative control. However, if the laboratory setup allows one also to remove the chemoattractant, or to add a repellent, the negative control becomes physically realizable. This can provide a better control of the cell population, whereas the low-rank numerical algorithms allow a fast simulation of the required profile of the attractant/repellent, even on a low performance desktop.

Acknowledgements. The authors are very grateful to two anonymous referees and the handling editor for their valuable comments, which greatly improved the manuscript. SD and JWP gratefully acknowledge support from the Engineering and Physical Sciences Research Council (UK) Fellowships EP/M019004/1 and EP/M018857/2, respectively. JWP also gratefully acknowledges a Fellowship from The Alan Turing Institute in London.

REFERENCES

[1] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll. Low-rank solvers for unsteady Stokes–Brinkman optimal control problem with random data. *Comput. Method. Appl. M.*, 304:26–54, 2016.
[2] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.
[3] E. Casas and F. Tröltzsch. Second order optimality conditions and their role in PDE control. *Jahresber. Deutsch. Math.-Verein.*, 117(1):3–44, 2015.
[4] J. C. De los Reyes and K. Kunisch. A semi-smooth Newton method for control constrained boundary optimal control of the Navier–Stokes equations. *Nonlin. Anal.-Theor.*, 62:1289–1316, 2005.
[5] V. de Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM J. Matrix Anal. Appl.*, 30(3):1084–1127, 2008.
[6] S. Dolgov, J. W. Pearson, D. V. Savostyanov, and M. Stoll. Fast tensor product solvers for optimization problems with fractional differential equations as constraints. *Appl. Math. Comput.*, 273:604–623, 2016.
[7] S. Dolgov and M. Stoll. Low-rank solution to an optimization problem constrained by the Navier–Stokes equations. *SIAM J. Sci. Comput.*, 39(1):A255–A280, 2017.
[8] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov. Computation of extreme eigenvalues in higher dimensions using block tensor train format. *Computer Phys. Comm.*, 185(4):1207–1216, 2014.
[9] H. Egger, J.-F. Pietschmann, and M. Schlottbom. Identification of chemotaxis models with volume-filling. *SIAM J. Appl. Math.*, 75(2):275–288, 2015.
[10] H. Feldhordt. *Boundary control of a chemotaxis system.* PhD thesis, Universität Duisburg–Essen, 2017.
[11] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods, Part I. *Numer. Math.*, 3:147–156, 1961.
[12] G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods, Part II. *Numer. Math.*, 3:157–168, 1961.
[13] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitt.*, 36(1):53–78, 2013.
[14] W. Hackbusch. *Tensor Spaces And Numerical Tensor Calculus.* Springer-Verlag, Berlin, 2012.
[15] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *J. Fourier Anal. Appl.*, 15(5):706–722, 2009.
[16] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM J. Optim.*, 13(3):865–888, 2003.
[17] M. Hintermüller and M. Ulbrich. A mesh-independence result for semismooth Newton methods. *Math. Program. (Ser. B)*, 101(1):151–184, 2004.
[18] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM J. Sci. Comput.*, 34(2):A683–A713, 2012.
[19] I. C. F. Ipsen. A note on preconditioning nonsymmetric matrices. *SIAM J. Sci. Comput.*, 23(3):1050–1051, 2001.
[20] K. Ito and K. Kunisch. Semi-smooth Newton methods for state-constrained optimal control problems. *Syst. Control Lett.*, 50:221–228, 2003.
[21] V. A. Kazeev and B. N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM J. Matrix Anal. Appl.*, 33(3):742–758, 2012.
[22] V. A. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov. Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity. *SIAM J. Sci. Comput.*, 35(3):A1511–A1536, 2013.
[23] E. F. Keller and L. A. Segel. Model for chemotaxis. *J. Theor. Biol.*, 30(2):225–234, 1971.
[24] B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of N–d tensors in high-dimensional numerical modeling. *Constr. Approx.*, 34(2):257–280, 2011.
[25] B. N. Khoromskij. Tensor numerical methods for multidimensional PDEs: Theoretical analysis and initial applications. *ESAIM: Proc.*, 48:1–28, 2015.
[26] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.
[27] Y. A. Kuznetsov. Efficient iterative solvers for elliptic finite element problems on nonmatching grids. *Russ. J. Numer. Anal. M.*, 10(3):187–212, 1995.
[28] D. Lebiedz and U. Brandt-Pollmann. Manipulation of self-aggregation patterns and waves in a reaction–diffusion system by optimal boundary control strategies. *Phys. Rev. Lett.*, 91(20):208301, 2003.

[29] D. Lebiedz and H. Maurer. External optimal control of self-organisation dynamics in a chemotaxis reaction diffusion system. *IEE P. Syst. Biol.*, 1(2):222–229, 2004.
[30] G. MacDonald, J. A. Mackenzie, M. Nolan, and R. H. Insall. A computational method for the coupled solution of reaction–diffusion equations on evolving domains and manifolds: Application to a model of cell migration and chemotaxis. *J. Comput. Phys.*, 309:207–226, 2016.
[31] A. Y. Mikhalev and I. V. Oseledets. Rectangular maximum-volume submatrices and their applications. *Linear Algebra Appl.*, 538:187–211, 2018.
[32] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.
[33] A. Napov and Y. Notay. An algebraic multigrid method with guaranteed convergence rate. *SIAM J. Sci. Comput.*, 34(2):A1079–A1109, 2012.
[34] Y. Notay. AGMG software and documentation; see `http://agmg.eu`.
[35] Y. Notay. An aggregation-based algebraic multigrid method. *Electron. Trans. Numer. Anal.*, 37(6):123–146, 2010.
[36] Y. Notay. Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM J. Sci. Comput.*, 34(4):A2288–A2316, 2012.
[37] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.
[38] I. V. Oseledets and E. E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.
[39] J. W. Pearson and J. Gondzio. Fast interior point solution of quadratic programming problems arising from PDE-constrained optimization. *Numer. Math.*, 137(4):959–999, 2017.
[40] J. W. Pearson and M. Stoll. Fast iterative solution of reaction–diffusion control problems arising from chemical processes. *SIAM J. Sci. Comput.*, 35(5):B987–B1009, 2013.
[41] J. W. Pearson, M. Stoll, and A. J. Wathen. Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems. *SIAM J. Matrix Anal. Appl.*, 33(4):1126–1152, 2012.
[42] J. W. Pearson, M. Stoll, and A. J. Wathen. Preconditioners for state-constrained optimal control problems with Moreau–Yosida penalty function. *Numer. Lin. Alg. Appl.*, 21(1):81–97, 2014.
[43] J. W. Pearson and A. J. Wathen. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numer. Lin. Alg. Appl.*, 19(5):816–829, 2012.
[44] J. W. Pearson and A. J. Wathen. Fast iterative solvers for convection–diffusion control problems. *Electron. Trans. Numer. Anal*, 40:294–310, 2013.
[45] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *Quantum Info. Comput.*, 7(5):401–430, 2007.
[46] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Universität Heidelberg, 2011.
[47] T. Rees, H. S. Dollar, and A. J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM J. Sci. Comput.*, 32(1):271–298, 2010.
[48] A. Rösch and D. Wachsmuth. Semi-smooth Newton method for an optimal control problem with control and mixed control-state constraints. *Optim. Method. Softw.*, 26(2):169–186, 2011.
[49] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.
[50] N. Saito. Conservative numerical schemes for the Keller–Segel system and numerical results. *RIMS Kôkyûroku Bessatsu*, B15:125–146, 2009.
[51] D. V. Savostyanov. Quasioptimality of maximum-volume cross interpolation of tensors. *Linear Algebra Appl.*, 458:217–244, 2014.
[52] D. V. Savostyanov and I. V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In *Proceedings of 7th International Workshop on Multidimensional Systems (nDS)*. IEEE, 2011.
[53] R. Schneider and A. Uschmajew. Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. *J. Complexity*, 30(2):56–71, 2014.
[54] J. Schöberl and W. Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM J. Matrix Anal. Appl.*, 29(3):752–773, 2007.
[55] U. Schollwöck. The density-matrix renormalization group. *Rev. Mod. Phys.*, 77(1):259–315, 2005.
[56] M. Stoll, J. W. Pearson, and P. K. Maini. Fast solvers for optimal control problems from pattern formation. *J. Comput. Phys.*, 304:27–45, 2016.

[57] R. Strehl, A. Sokolov, D. Kuzmin, D. Horstmann, and S. Turek. A positivity-preserving finite element method for chemotaxis problems in 3D. *J. Comput. Appl. Math.*, 239:290–303, 2013.

[58] R. Strehl, A. Sokolov, and S. Turek. Efficient, accurate and flexible finite element solvers for chemotaxis problems. *J. Comput. Phys.*, 64(3):175–189, 2012.

[59] E. E. Tyrtyshnikov. Tensor approximations of matrices generated by asymptotically smooth functions. *Sb. Math.*, 194(6):941–954, 2003.

[60] R. Tyson, S. R. Lubkin, and J. D. Murray. A minimal mechanism for bacterial pattern formation. *P. Roy. Soc. B – Biol. Sci.*, 266(1416):299–304, 1999.

[61] R. Tyson, S. R. Lubkin, and J. D. Murray. Model and analysis of chemotactic bacterial patterns in a liquid medium. *J. Math. Biol.*, 38(4):359–375, 1999.

[62] R. Tyson, L. G. Stern, and R. J. LeVeque. Fractional step methods applied to a chemotaxis model. *J. Math. Biol.*, 41:455–475, 2000.

[63] M. Ulbrich. Semismooth Newton methods for operator equations in function spaces. *SIAM J. Optim.*, 13(1):805–841, 2003.

[64] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

[65] A. J. Wathen. Realistic eigenvalue bounds for the Galerkin mass matrix. *IMA J. Numer. Anal.*, 7(4):449–457, 1987.

[66] A. J. Wathen and T. Rees. Chebyshev semi-iteration in preconditioning for problems including the mass matrix. *Electron. Trans. Numer. Anal.*, 34:125–135, 2009.

[67] W. Zulehner. Nonstandard norms and robust estimates for saddle point problems. *SIAM J. Matrix Anal. Appl.*, 32(2):536–560, 2011.